Objektorientierte Programmierung mit C++ (WS 2010)

Dr. Andreas F. Borchert, Tobias Brosch
Institut für Angewandte Informationsverarbeitung
Universität Ulm

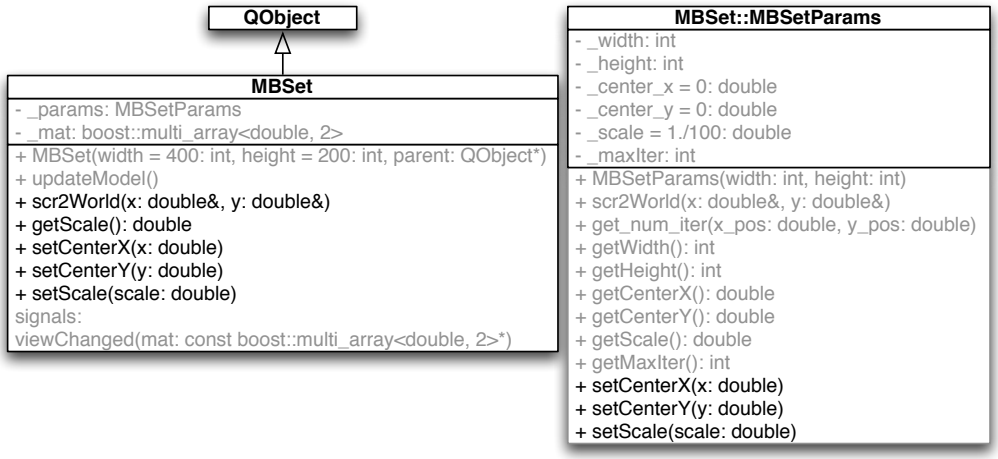
Blatt 12: Abgabetermin 26. Januar 2011 11 Uhr

Mandelbrotmenge Interaktiv (Teil 2)

Unseren Mandelbrot-Viewer aus dem letzten Blatt werden wir nun dahingehend erweitern, dass ein Linksklick mit der Maus an die entsprechende Stelle hineinzoomt, und ein Rechtsklick an der entsprechenden Stelle hinauszoomt.

Model

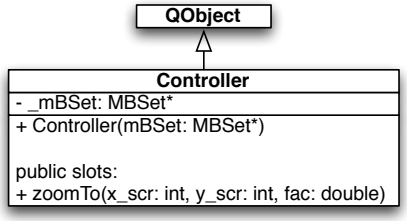
Erweitert die Klasse `MBSet` um folgende hervorgehobenen Methoden:



Dabei soll die Klasse `MBSet` die entsprechenden Aufrufe lediglich an ihr `_params`-Objekt delegieren.

Controller

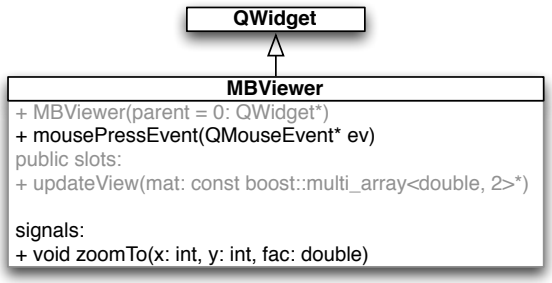
Implementiert die `Controller`-Klasse wie folgt:



Die Parameter `x_scr` und `y_scr` der Methode `zoomTo` werden später dann die Pixelkoordinaten sein, auf die der Nutzer geklickt hat. Der Faktor `fac` wird bei einem Rechtsklick als 2 übergeben und bei einem Linksklick als 0.5 (das soll jedoch die View entscheiden). Die Methode `zoomTo` soll dann Unsere ergänzten Methoden in `MBSet` nutzen, um das Zentrum auf die geklickten Koordinaten zu setzen (natürlich mit `scr2World` in Koordinaten der komplexen Ebene umgerechnet) und die Skalierung um den gegebenen Faktor multipliziert und setzt (vergesst nicht das `Q_OBJECT`-Makro!). Final soll die `updateModel()`-Methode aufgerufen werden, um das Modell mit den neuen Parametern zu aktualisieren.

View

Nun muss die View noch das passende Signal zu der Methode `zoomTo` aussenden. Dazu definieren wir zunächst das entsprechende Signal und reimplementieren die geerbte Methode `mousePressEvent`, welche einen Zeiger auf ein `QMouseEvent` übergeben bekommt.



Wie bereits beschrieben, senden wir hierbei jeweils je nach Mausklick ein Signal, welches (nach dem Verknüpfen in `main`) unsere Methode `zoomTo` des Controllers aufruft:

```
void MBViewer::mousePressEvent(QMouseEvent* ev) {
        QPoint point = ev->pos();
        double x = point.x();
        double y = point.y();
        if(ev->button() == Qt::LeftButton) {
                emit zoomTo(x, y, .5);
```

```
        } else if(ev->button() == Qt::RightButton) {
                emit zoomTo(x, y, 2);
        }
}
```

Verbinden und Eintauchen in die Welt der Mandelbrotmenge

Alles was jetzt noch fehlt, ist in der Methode `main` ein Controller-Objekt `controller` anzulegen und das Signal von `mBViewer` mit dem Slot von `controller` zu verbinden.

Viel Spaß beim Erkunden des Apfelmännchens!

Beachtet insbesondere, dass je nach Anzahl der maximalen Iterationen, früher oder später die Ränder der Mandelbrotmenge pixelig oder ungewöhnlich "glatt" werden. Wer sich übrigens daran stört, dass jeder Mausklick auch eine Berechnung zur Folge hat (die je nach Größe des Bildes durchaus dauern kann und bei vier Mausklicks benötigt der User nicht unbedingt jedes Zwischenresultat), der kann sich auf das nächste Blatt freuen, denn hier werden Wir sehen, wie die Berechnung in einen eigenen Thread ausgelagert werden kann.

Submission

Einreichen der Lösung mit:

```
submit cpp 12 main.cpp MBSet.h MBSet.cpp MBViewer.h MBViewer.cpp \
        Controller.h Controller.cpp
```

Viel Spaß!